EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
		("6542933").PN.	USPAT; USOCR	OR	OFF	2004/07/02 09:19
L1	13	(virtual adj component$1) same identif$4	USPAT	OR	ON	2006/09/26 09:29
L2	101	((virtual or logical)adj component$1) same identif$4	USPAT	OR	ON	2006/09/26 09:31
L3	46	((virtual or logical)adj component$1) with identif$4	USPAT	OR	ON	2006/09/26 09:31
L4	0	((virtual or logical)adj component$1) with identif$4 same decompos$5 same node	USPAT	OR	ON	2006/09/26 09:31
L5	0	((virtual or logical)adj component$1) with identif$4 same decompose$5 same node	USPAT	OR	ON	2006/09/26 09:32
L6	0	((virtual or logical)adj component$1) with identif$4 same decompose same node$1	USPAT	OR	ON	2006/09/26 09:32
L7	166970	((virtual or logical)adj component$1) with identif$4same node$1	USPAT	OR	ON	2006/09/26 09:32
L8	5	((virtual or logical)adj component$1) with identif$4 same node$1	USPAT	OR	ON	2006/09/26 09:32
L9	0	((virtual or logical)adj component$1) with identif$4 same node$1 same remove$ same redundant	USPAT	OR	ON	2006/09/26 09:33
L10	0	((virtual or logical)adj component$1) with identif$4 same node$1 same redundant	USPAT	OR	ON	2006/09/26 09:34
L11	0	((virtual or logical) adj component$1) with identif$4 same node$1 same redundant	USPAT	OR	ON	2006/09/26 09:35
L12	0	((virtual or logical) adj component$1) same identif$4 same node$1 same redundant	USPAT	OR	ON	2006/09/26 09:35
L13	0	((virtual or logical) adj component$1) same identif$4 same node$1 same (email or e-mail or (e adj mail))	USPAT	OR	ON	2006/09/26 09:36
L14	3	((virtual or logical) adj component$1) same identif$4 same node$1 and (email or e-mail or (e adj mail))	USPAT	OR	ON	2006/09/26 09:36
S1	1	email adj relationship	USPAT	OR	ON	2004/06/04 15:41

EAST Search History

S2	0	email adj threads	USPAT	OR	ON	2004/06/04 15:43
S3	125	message adj threads	USPAT	OR	ON	2004/06/04 15:43
S4	8	(message adj threads) same relationship	USPAT	OR	ON	2004/06/04 15:51
S5	95	(message adj threads) and generate$3	USPAT	OR	ON	2004/06/04 15:52
S6	0	(message adj threads) and (generate$3 adj document)	USPAT	OR	ON	2004/06/04 15:53
S7	18	(message adj threads) and (generate$3 adj message)	USPAT	OR	ON	2004/06/04 15:53
S8	9	(message adj threads) and (generate$3 adj message) and relationship	USPAT	OR	ON	2004/06/04 16:04
S9	905	709/206.ccls.	USPAT	OR	ON	2004/06/04 16:04
S10	4	((message adj threads) and (generate$3 adj message) and relationship) and 709/206.ccls.	USPAT	OR	ON	2004/06/07 09:53
S11	0	(((e adj mail) or e-mail or email)adj threads) and link$3 and email adj chain	USPAT	OR	ON	2004/06/07 09:54
S12	0	(((e adj mail) or e-mail or email)adj threads) and link$3 and (email adj chain)	USPAT	OR	ON	2004/06/07 09:55
S13	0	(((e adj mail) or e-mail or email)adj threads) and (email adj chain)	USPAT	OR	ON	2004/06/07 09:55
S14	11	(((e adj mail) or e-mail or email)adj threads)	USPAT	OR	ON	2004/06/07 09:56
S15	0	(((e adj mail) or e-mail or email)adj threads) and chain	USPAT	OR	ON	2004/06/07 09:55
S16	1	(((e adj mail) or e-mail or email)adj chain) and link	USPAT	OR	ON	2004/06/07 09:55
S17	10	(((e adj mail) or e-mail or email)adj threads)and link	USPAT	OR	ON	2004/06/07 09:56
S18	10	(((e adj mail) or e-mail or email)adj threads) and link	USPAT	OR	ON	2004/06/07 09:56
S19	10	(((e adj mail) or e-mail or email)adj threads) and link and generate$3	USPAT	OR	ON	2004/06/07 16:31
S20	20	(((e adj mail) or e-mail or email)adj relat$4) and link	USPAT	OR	ON	2004/06/07 16:32
S21	1	(((e adj mail) or e-mail or email)adj relat$4) and threads	USPAT	OR	ON	2004/06/10 17:34
S22	71	(relate$4 adj ((e adj mail) or e-mail or email))	USPAT	OR	ON	2004/06/10 17:35

EAST Search History

S23	0	(relate$4 adj ((e adj mail) or e-mail or email)) same generate$4 adj document	USPAT	OR	ON	2004/06/10 17:35
S24	0	(relate$4 adj ((e adj mail) or e-mail or email)) and (generate$4 adj document)	USPAT	OR	ON	2004/06/10 17:35
S25	49	(relate$4 adj ((e adj mail) or e-mail or email)) and (generate$4)	USPAT	OR	ON	2004/06/10 17:36
S26	1	(relate$4 adj ((e adj mail) or e-mail or email)) and (document adj generate$4)	USPAT	OR	ON	2004/06/10 17:36
S27	0	(((e adj mail) or e-mail or email) adj relate$4) and (document adj generate$4)	USPAT	OR	ON	2004/06/10 17:36
S28	27	(((e adj mail) or e-mail or email) adj relate$4)	USPAT	OR	ON	2004/06/10 17:36
S29	1659	((((e adj mail) or e-mail or email) or message)adj relate$4)	USPAT	OR	ON	2004/06/10 17:37
S30	8	((((e adj mail) or e-mail or email) or message)adj relate$4) and (generate$4 adj document)	USPAT	OR	ON	2004/06/10 17:39
S31	0	(((e adj mail) or e-mail or email)adj relate$4) and (generate$4 adj document)	USPAT	OR	ON	2004/06/10 17:39
S32	27	(((e adj mail) or e-mail or email)adj relate$4)	USPAT	OR	ON	2004/06/10 17:39
S33	0	(((e adj mail) or e-mail or email)adj relate$4) same threads	USPAT	OR	ON	2004/06/10 17:40
S34	3	(((e adj mail) or e-mail or email)adj relate$4) same generate$4	USPAT	OR	ON	2004/06/10 17:40
S35	19	(((e adj mail) or e-mail or email)adj relate$4) and generate$4	USPAT	OR	ON	2004/06/11 19:00
S36	46	(similar adj ((e adj mail) or e-mail or email)) and generate$4	USPAT	OR	ON	2004/06/11 19:00
S37	0	(relevent adj ((e adj mail) or e-mail or email)) and generate$4	USPAT	OR	ON	2004/06/11 19:01
S38	15	(similar adj ((e adj mail) or e-mail or email)) and threads	USPAT	OR	ON	2004/06/11 19:02
S39	5	(similar adj ((e adj mail) or e-mail or email)) with address	USPAT	OR	ON	2004/06/11 19:03
S40	0	(relavent adj ((e adj mail) or e-mail or email)) with address	USPAT	OR	ON	2004/06/11 19:03
S41	0	(relavent adj ((e adj mail) or e-mail or email))	USPAT	OR	ON	2004/06/11 19:03

EAST Search History

S42	0	(relevent adj ((e adj mail) or e-mail or email))	USPAT	OR	ON	2004/06/11 19:03
S43	0	(irrelevent adj ((e adj mail) or e-mail or email))	USPAT	OR	ON	2004/06/11 19:04
S44	2	(irrelevant adj ((e adj mail) or e-mail or email))	USPAT	OR	ON	2004/06/11 19:04
S45	12	(relevant adj ((e adj mail) or e-mail or email))	USPAT	OR	ON	2004/06/11 19:04
S46	3	(relevant adj ((e adj mail) or e-mail or email)) and threads	USPAT	OR	ON	2004/06/27 16:01
S47	8	(relevant adj ((e adj mail) or e-mail or email)) and generate$4	USPAT	OR	ON	2004/06/11 19:04
S48	1	(relevant adj ((e adj mail) or e-mail or email)) and (generate$4 adj document)	USPAT	OR	ON	2004/06/11 19:05
S49	4	(relevant adj ((e adj mail) or e-mail or email)) and generate$4 same document	USPAT	OR	ON	2004/06/11 19:07
S50	6113	generat$4 adj determin$4	USPAT	OR	ON	2004/06/27 15:56
S51	0	generat$4 adj determin$4 adj correlate$3	USPAT	OR	ON	2004/06/27 15:56
S52	3	generat$4 adj determin$4 adj relate$4	USPAT	OR	ON	2004/06/27 16:00
S53	1	("6484196").PN.	USPAT; USOCR	OR	OFF	2004/06/27 16:00
S54	0	(relevant adj ((e adj mail) or e-mail or email)) and weight	USPAT	OR	ON	2004/06/27 16:01
S55	2	(relevant adj ((e adj mail) or e-mail or email)) and path	USPAT	OR	ON	2004/06/27 16:08
S56	1	("6145079").PN.	USPAT; USOCR	OR	OFF	2004/06/27 16:03
S57	1	("6525747").PN.	USPAT; USOCR	OR	OFF	2004/06/27 16:03
S58	145	weight adj path	USPAT	OR	ON	2004/06/27 16:09
S59	0	(weight adj path) and ((e adj mail) or email or e-mail) and redundant	USPAT	OR	ON	2004/06/27 16:09
S60	5	(weight adj path) and ((e adj mail) or email or e-mail)	USPAT	OR	ON	2004/06/27 16:11
S61	175	identical same ((e adj mail) or email or e-mail)	USPAT	OR	ON	2004/06/27 16:13
S62	4	(redundant and identical) same ((e adj mail) or email or e-mail)	USPAT	OR	ON	2004/06/27 16:14

EAST Search History

S63	1	(redundant and identical) same ((e adj mail) or email or e-mail) and path	USPAT	OR	ON	2004/06/27 16:16
S64	1	(redundant and identical) same ((e adj mail) or email or e-mail) and max$4 and path	USPAT	OR	ON	2004/06/29 14:13
S65	1	"5206934".PN.	USPAT	OR	OFF	2004/06/27 16:26
S66	4	brendel.in. and load adj balanc$6	USPAT	OR	ON	2004/06/29 14:20
S67	199	(load adj balanc$6) and (server adj determine$4)	USPAT	OR	ON	2004/06/29 14:20
S68	10	(server adj determine$4) with (load adj balanc$6)	USPAT	OR	ON	2004/07/01 15:37
S69	7	(("5,845,215") or ("5,856,981") or ("5,999,103") or ("6,016,307") or ("6,295,275") or ("6,314,093") or ("6,411,946")).PN.	USPAT; USOCR	OR	OFF	2004/07/02 09:19

PORTAL USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: The ACM Digital Library The Guide

identical same logical same component same email same threa SEARCH

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **identical same logical same component same email same thread same redundant**

Found **124,940** of **185,942**

Sort results by relevance

Display results expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Fast detection of communication patterns in distributed executions

Thomas Kunz, Michiel F. H. Seuren

November 1997 **Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research**

Publisher: IBM Press

Full text available: pdf(4.21 MB) Additional Information: full citation, abstract, references, index terms

Understanding distributed applications is a tedious and difficult task. Visualizations based on process-time diagrams are often used to obtain a better understanding of the execution of the application. The visualization tool we use is Poet, an event tracer developed at the University of Waterloo. However, these diagrams are often very complex and do not provide the user with the desired overview of the application. In our experience, such tools display repeated occurrences of non-trivial commun ...

2 Efficient logic variables for distributed computing

Seif Haridi, Peter Van Roy, Per Brand, Michael Mehl, Ralf Scheidhauer, Gert Smolka

May 1999 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 21 Issue 3

Publisher: ACM Press

Full text available: pdf(572.35 KB) Additional Information: full citation, abstract, references, citings, index terms

We define a practical algorithm for distrubuted rational tree unification and prove its correctness in both the off-line and on-line cases. We derive the distributed algorithm from a centralized one, showing clearly the trade-offs between local and distributed execution. The algorithm is used to realize logic variables in the Mozart Programming System, which implements the Oz language (see http://www/mozart-oz.org). Oz appears to the programmer as a concurrent object-oriented language with ...

Keywords: Mozart, Oz, distributed algorithms

3 Real-time shading

Marc Olano, Kurt Akeley, John C. Hart, Wolfgang Heidrich, Michael McCool, Jason L. Mitchell, Randi Rost

August 2004 **Proceedings of the conference on SIGGRAPH 2004 course notes SIGGRAPH '04**

Publisher: ACM Press

Full text available: pdf(7.39 MB) Additional Information: full citation, abstract

Real-time procedural shading was once seen as a distant dream. When the first version of this course was offered four years ago, real-time shading was possible, but only with one-

of-a-kind hardware or by combining the effects of tens to hundreds of rendering passes. Today, almost every new computer comes with graphics hardware capable of interactively executing shaders of thousands to tens of thousands of instructions. This course has been redesigned to address today's real-time shading capabili ...

4 GPGPU: general purpose computation on graphics hardware

David Luebke, Mark Harris, Jens Krüger, Tim Purcell, Naga Govindaraju, Ian Buck, Cliff Woolley, Aaron Lefohn

August 2004 **Proceedings of the conference on SIGGRAPH 2004 course notes SIGGRAPH '04**

Publisher: ACM Press

Full text available: pdf(63.03 MB) Additional Information: full citation, abstract

The graphics processor (GPU) on today's commodity video cards has evolved into an extremely powerful and flexible processor. The latest graphics architectures provide tremendous memory bandwidth and computational horsepower, with fully programmable vertex and pixel processing units that support vector operations up to full IEEE floating point precision. High level languages have emerged for graphics hardware, making this computational power accessible. Architecturally, GPUs are highly parallel s ...

5 Verification techniques for cache coherence protocols

Fong Pong, Michel Dubois

March 1997 **ACM Computing Surveys (CSUR)**, Volume 29 Issue 1

Publisher: ACM Press

Full text available: pdf(1.25 MB) Additional Information: full citation, abstract, references, citings, index terms

In this article we present a comprehensive survey of various approaches for the verification of cache coherence protocols based on state enumeration, (symbolic model checking, and symbolic state models. Since these techniques search the state space of the protocol exhaustively, the amount of memory required to manipulate that state information and the verification time grow very fast with the number of processors and the complexity of the protocol mechanism ...

Keywords: cache coherence, finite state machine, protocol verification, shared-memory multiprocessors, state representation and expansion

6 Simplify: a theorem prover for program checking

David Detlefs, Greg Nelson, James B. Saxe

May 2005 **Journal of the ACM (JACM)**, Volume 52 Issue 3

Publisher: ACM Press

Full text available: pdf(1.93 MB) Additional Information: full citation, abstract, references, index terms

This article provides a detailed description of the automatic theorem prover Simplify, which is the proof engine of the Extended Static Checkers ESC/Java and ESC/Modula-3. Simplify uses the Nelson--Oppen method to combine decision procedures for several important theories, and also employs a matcher to reason about quantifiers. Instead of conventional matching in a term DAG, Simplify matches up to equivalence in an E-graph, which detects many relevant pattern instances that would be missed by th ...

Keywords: Theorem proving, decision procedures, program checking

7 Run-time adaptation in river

Remzi H. Arpaci-Dusseau

February 2003 **ACM Transactions on Computer Systems (TOCS)**, Volume 21 Issue 1

Publisher: ACM Press

Full text available: pdf(849.04 KB) Additional Information: full citation, abstract, references, index terms

We present the design, implementation, and evaluation of run-time adaptation within the River dataflow programming environment. The goal of the River system is to provide adaptive mechanisms that allow database query-processing applications to cope with performance variations that are common in cluster platforms. We describe the system and its basic mechanisms, and carefully evaluate those mechanisms and their effectiveness. In our analysis, we answer four previously unanswered and important que ...

Keywords: Performance availability, clusters, parallel I/O, performance faults, robust performance, run-time adaptation

8 System-level power optimization: techniques and tools

Luca Benini, Giovanni de Micheli
April 2000 **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, Volume 5 Issue 2
Publisher: ACM Press

Full text available: pdf(385.22 KB) Additional Information: full citation, abstract, references, citings, index terms

This tutorial surveys design methods for energy-efficient system-level design. We consider electronic sytems consisting of a hardware platform and software layers. We consider the three major constituents of hardware that consume energy, namely computation, communication, and storage units, and we review methods of reducing their energy consumption. We also study models for analyzing the energy cost of software, and methods for energy-efficient software design and compilation. This survery ...

9 Toward a logical/physical theory of spreadsheet modeling

Tomás Isakowitz, Shimon Schocken, Henry C. Lucas
January 1995 **ACM Transactions on Information Systems (TOIS)**, Volume 13 Issue 1
Publisher: ACM Press

Full text available: pdf(2.76 MB) Additional Information: full citation, abstract, references, citings, index terms, review

In spite of the increasing sophistication and power of commercial spreadsheet packages, we still lack a formal theory or a methodology to support the construction and maintenance of spreadsheet models. Using a dual logical/physical perspective, we identify four principal components that characterize any spread sheet model: schema, data, editorial, and binding. We present a factoring algorithm for identifying and extracting these components ...

Keywords: model management

10 HFS: a performance-oriented flexible file system based on building-block compositions

Orran Krieger, Michael Stumm
August 1997 **ACM Transactions on Computer Systems (TOCS)**, Volume 15 Issue 3
Publisher: ACM Press

Full text available: pdf(383.87 KB) Additional Information: full citation, abstract, references, citings, index terms, review

The Hurricane File System (HFS) is designed for (potentially large-scale) shared-memory multiprocessors. Its architecture is based on the principle that, in order to maximize performance for applications with diverse requirements, a file system must support a wide variety of file structures, file system policies, and I/O interfaces. Files in HFS are implemented using simple building blocks composed in potentially complex ways. This approach yields great flexibility, allowing an application ...

Keywords: customization, data partitioning, data replication, flexibility, parallel computing, parallel file system

11 Symbolic bounds analysis of pointers, array indices, and accessed memory regions

Radu Rugina, Martin C. Rinard

March 2005 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 27 Issue 2

Publisher: ACM Press

Full text available: pdf(490.56 KB) Additional Information: full citation, abstract, references, index terms

This article presents a novel framework for the symbolic bounds analysis of pointers, array indices, and accessed memory regions. Our framework formulates each analysis problem as a system of inequality constraints between symbolic bound polynomials. It then reduces the constraint system to a linear program. The solution to the linear program provides symbolic lower and upper bounds for the values of pointer and array index variables and for the regions of memory that each statement and procedur ...

Keywords: Symbolic analysis, parallelization, static race detection

12 Cloning-based context-sensitive pointer alias analysis using binary decision diagrams

John Whaley, Monica S. Lam

June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation PLDI '04**, Volume 39 Issue 6

Publisher: ACM Press

Full text available: pdf(277.87 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents the first scalable context-sensitive, inclusion-based pointer alias analysis for Java programs. Our approach to context sensitivity is to create a clone of a method for every context of interest, and run a *context-insensitive* algorithm over the expanded call graph to get *context-sensitive* results. For precision, we generate a clone for every acyclic path through a program's call graph, treating methods in a strongly connected component as a single node. Normally ...

Keywords: Datalog, Java, binary decision diagrams, cloning, context-sensitive, inclusion-based, logic programming, pointer analysis, program analysis, scalable

13 Coyote: a system for constructing fine-grain configurable communication services

Nina T. Bhatti, Matti A. Hiltunen, Richard D. Schlichting, Wanda Chiu

November 1998 **ACM Transactions on Computer Systems (TOCS)**, Volume 16 Issue 4

Publisher: ACM Press

Full text available: pdf(290.21 KB) Additional Information: full citation, abstract, references, citings, index terms

Communication-oriented abstractions such as atomic multicast, group RPC, and protocols for location-independent mobile computing can simplify the development of complex applications built on distributed systems. This article describes Coyote, a system that supports the construction of highly modular and configurable versions of such abstractions. Coyote extends the notion of protocol objects and hierarchical composition found in existing systems with support for finer-grain microprotocol ob ...

Keywords: x-kernal, configurable sevices, customization, event handlers, event-driven execution, membership, microprotocols, mobile computing, modularity, multicast, protocols, remote procedure call

14 Tutorial: Compiling concurrent languages for sequential processors

Stephen A. Edwards

April 2003 **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, Volume 8 Issue 2

Publisher: ACM Press

Full text available: pdf(771.65 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Embedded systems often include a traditional processor capable of executing sequential code, but both control and data-dominated tasks are often more naturally expressed using one of the many domain-specific concurrent specification languages. This article surveys a variety of techniques for translating these concurrent specifications into sequential code. The techniques address compiling a wide variety of languages, ranging from dataflow to Petri nets. Each uses a different method, to some degr ...

Keywords: Compilation, Esterel, Lustre, Petri nets, Verilog, code generation, communication, concurrency, dataflow, discrete-event, partial evaluation, sequential

15 The Vesta parallel file system

Peter F. Corbett, Dror G. Feitelson

August 1996 **ACM Transactions on Computer Systems (TOCS)**, Volume 14 Issue 3

Publisher: ACM Press

Full text available: pdf(649.08 KB) Additional Information: full citation, abstract, references, citings, index terms, review

The Vesta parallel file system is designed to provide parallel file access to application programs running on multicomputers with parallel I/O subsystems. Vesta uses a new abstraction of files: a file is not a sequence of bytes, but rather it can be partitioned into multiple disjoint sequences that are accessed in parallel. The partitioning—which can also be changed dynamically—reduces the need for synchronization and coordination during the access. Some control over the layout ...

Keywords: data partitioning, parallel computing, parallel file system

16 A concurrency analysis tool suite for Ada programs: rationale, design, and preliminary experience

Michal Young, Richard N. Taylor, David L. Levine, Kari A. Nies, Debra Brodbeck

January 1995 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 4 Issue 1

Publisher: ACM Press

Full text available: pdf(2.93 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Cats (Concurrency Analysis Tool Suite) is designed to satisfy several criteria: it must analyze implementation-level Ada source code and check user-specified conditions associated with program source code; it must be modularized in a fashion that supports flexible composition with other tool components, including integration with a variety of testing and analysis techniques; and its performance and capacity must be sufficient for analysis of real application programs. Meeting these objectiv ...

Keywords: Ada, concurrency, software development environments, static analysis, tool integration

17 The KaffeOS Java runtime system

Godmar Back, Wilson C. Hsieh

July 2005 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 27 Issue 4

Publisher: ACM Press

Full text available: pdf(704.30 KB) Additional Information: full citation, abstract, references, index terms

Single-language runtime systems, in the form of Java virtual machines, are widely deployed platforms for executing untrusted mobile code. These runtimes provide some of the features that operating systems provide: interapplication memory protection and basic

system services. They do not, however, provide the ability to isolate applications from each other. Neither do they provide the ability to limit the resource consumption of applications. Consequently, the performance of current systems degra ...

Keywords: Robustness, garbage collection, isolation, language runtimes, resource management, termination, virtual machines

18 An abstract machine for tabled execution of fixed-order stratified logic programs

Konstantinos Sagonas, Terrance Swift
May 1998 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 20 Issue 3
Publisher: ACM Press

Full text available: pdf(602.38 KB) Additional Information: full citation, abstract, references, citings, index terms, review

SLG resolution uses tabling to evaluate nonfloundering normal logic pr ograms according to the well-founded semantics. The SLG-WAM, which forms the engine of the XSB system, can compute in-memory recursive queries an order of magnitute faster than current deductive databases. At the same time, the SLG-WAM tightly intergrates Prolog code with tabled SLG code, and executes Prolog code with minimal overhead compared to the WAM. As a result, the SLG-WAM brings to logic progr ...

Keywords: SLG, WAM, memoing, prolog, stratification theories, tabling

19 Equal rights for functional objects or, the more things change, the more they are the same

Henry G. Baker
October 1993 **ACM SIGPLAN OOPS Messenger**, Volume 4 Issue 4
Publisher: ACM Press

Full text available: pdf(2.61 MB) Additional Information: full citation, abstract, index terms

We argue that intensional *object identity* in object-oriented programming languages and databases is best defined operationally by side-effect semantics. A corollary is that "functional" objects have extensional semantics. This model of object identity, which is analogous to the normal forms of relational algebra, provides cleaner semantics for the value-transmission operations and built-in primitive equality predicate of a programming language, and eliminates the confusion surrounding "ca ...

20 A general framework for prefetch scheduling in linked data structures and its application to multi-chain prefetching

Seungryul Choi, Nicholas Kohout, Sumit Pamnani, Dongkeun Kim, Donald Yeung
May 2004 **ACM Transactions on Computer Systems (TOCS)**, Volume 22 Issue 2
Publisher: ACM Press

Full text available: pdf(2.45 MB) Additional Information: full citation, abstract, references, index terms

Pointer-chasing applications tend to traverse composite data structures consisting of multiple independent pointer chains. While the traversal of any single pointer chain leads to the serialization of memory operations, the traversal of independent pointer chains provides a source of memory parallelism. This article investigates exploiting such *interchain memory parallelism* for the purpose of memory latency tolerance, using a technique called *multi--chain prefetching*. Previous work ...

Keywords: Data prefetching, memory parallelism, pointer-chasing code

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2006 ACM, Inc.

Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player